



MAKING AN AMAZON ECHO COMPATIBLE LINUX SYSTEM

Mike Anderson

Chief Scientist

The PTR Group, Inc.

<http://ThePTRGroup.com>

mike@theptrgroup.com

Copyright 2017, The PTR Group, Inc.

Who is The PTR Group?

- ✱ The PTR Group was founded in 2000
- ✱ We are involved in multiple areas of work:
 - ▶ Robotics (NASA space arm)
 - ▶ Flight software (over 35 satellites on orbit)
 - ▶ Offensive and defensive cyber operations
 - I'll leave this to your imagination ☺
 - ▶ Embedded software ports to RTOS/Linux/bare metal
 - ▶ IoT systems architecture and deployment

Speaker/Author Details



- Website:
 - <http://www.theptrgroup.com>
- Email:
 - <mailto:mike@theptrgroup.com>
- Linked-in:
 - <https://www.linkedin.com/in/mikeandersonptr>
- Twitter:
 - @hungjar

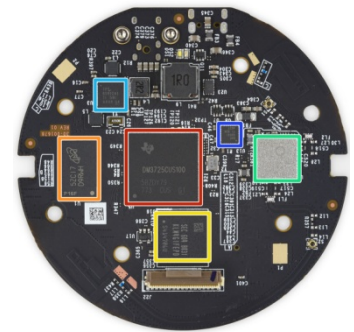
Almost 40 years in the embedded and real-time industry for both commercial and Government customers.

What We'll Talk About...

- ✦ Why do this?
- ✦ How Amazon's Echo works
- ✦ Alexa Voice Services
- ✦ Setting up the system
- ✦ Registering with AVS
- ✦ Getting an application running
- ✦ Summary

Why do this?

- ✖ The simple answer is because I thought it was cool 😊
- ✖ After seeing an Amazon Echo Dot, I wondered how the system worked
 - ▶ So, I started researching it and found a teardown of the unit on iFixit
 - It's basically an Beaglebone with considerable effort put towards audio processing clean up
- ✖ It looked simple enough, so I thought I'd try to make one



Source: iFixit.com

What does Echo do?

- ✦ Amazon's Echo system comes in two different versions
 - ▶ The standard Echo that comes with a built in speaker
 - ▶ The Echo Dot that's sans the big speaker
- ✦ Both use Alexa Voice Services (AVS) to process the audio and turn it into commands for the system
- ✦ There are several devices that are AVS enabled including Nest thermostats, Wemo plugs and Philips Hue Lightbulbs
- ✦ This is in addition to answering questions, ordering pizza delivery, calling for Uber, streaming media and a bunch of other things
 - ▶ So, it is a digital assistant that I didn't know I needed 😊



Source: amazon.com



Source: amazon.com

How does it Work?

- ✱ As a developer, you take your device and add voice recognition using AVS APIs
 - ▶ Core functionality like audio playback, volume control and text-to-speech are already available
- ✱ Your device has to have a microphone, speaker and an Internet connection
- ✱ The device waits for the “wake word” – in this case “Alexa” and then records what you say and send it to the Amazon cloud for processing
 - ▶ Be aware that the device is constantly listening and waiting for the wake word
 - Yes, you can change the wake word

How does it Work? (2)

- ✱ Once AWS has it figured out, an “Alexa Skill” is invoked for the device and the result is sent back to the device as a message
 - ▶ There are 1000’s of these skills already defined or you can build your own
- ✱ Your device then implements the skill

Where does it work?

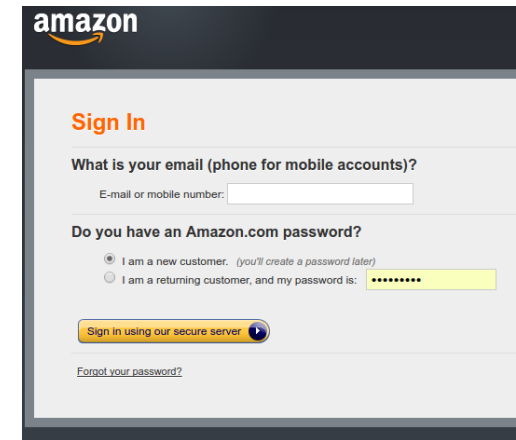
- ✱ AVS was originally launched in the US
 - ▶ The speech engines were tuned for American English
- ✱ Recently, Amazon announced support for both UK English and German languages
 - ▶ You need to specify what region of the world you are in so you get Uber instead of MyTaxi and Dominoes instead of JustEat, etc.
- ✱ Each additional region will need to have its own speech engine built
 - ▶ Fortunately, the skills and the API are just code and won't need to change

Design Guidelines

- ✱ Before you try to add voice recognition to project, you need to think through the interaction use cases
 - ▶ Will your device actually benefit from voice activation?
- ✱ There are also several Automatic Speech Recognition (ASR) profiles that are available for use
 - ▶ Use the distance to the microphone and background noise levels to select an ASR (near-field vs. far-field audio)
- ✱ The ASR then provides filtering to Alexa's Natural Language Understanding (NLU) engine to determine the command and send your device the appropriate message

How to get Started?

- ✱ First, sign up for an Amazon developer account at:
<https://developer.amazon.com/login.html>
 - ▶ The account is free
- ✱ Next, start learning about the Alexa Skills Kit (ASK) and how to use AVS to invoke the skills
 - ▶ <https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/getting-started-guide>
- ✱ There are several examples on the Amazon developer's site



Source: amazon.com

Getting Started with the Alexa Skills Kit

Learn how to give Alexa new abilities with the Alexa Skills Kit.

- Overview
- What Kind of Skill Do You Want to Create?
- What Do I Build When Creating a Skill?
- Next Steps

Source: amazon.com

Writing the Cloud Part

- ✱ All of your Alexa skills will be executed in Amazon's Web Service (AWS)
 - ▶ You're not executing in a VM, but rather using a different service targeted at remote procedure call-like features
- ✱ You can write your skills in either Java or Node.js
 - ▶ There's github repo for the Node.js sdk:
 - <https://github.com/alexa/alexa-skills-kit-sdk-for-nodejs>

What You will Need in Hardware

- ✱ The Echo dot is really just a collection of microphones, a speaker and a small computer with to access the Internet
 - ▶ Based on TI DM3725 ARM Cortex A8
- ✱ So, most of the current class of ARM development boards will work
- ✱ You'll need an Internet connection, microphone and speaker
 - ▶ A USB microphone and an audio amplifier with speaker will do
- ✱ I used a Raspberry Pi for the first implementation because it already had Wi-Fi and audio out
- ✱ Plus the usual power supply, Ethernet, SD card, etc. to get the OS (Raspbian) up and running



Source: amazon.com

Setting up the Raspberry Pi

✱ This is just the normal process for setting up any of the development boards

- ▶ Download Debian Jessie (Raspbian), copy it to an SD card using dd and boot the device
- ▶ <https://www.raspberrypi.org/downloads/raspbian/>

✱ I used this really cool laptop version that I built on top of a Raspberry Pi 3

- ▶ <https://www.pi-top.com/product/pi-top>
 - Naturally, getting ALSA up and running was the tough part



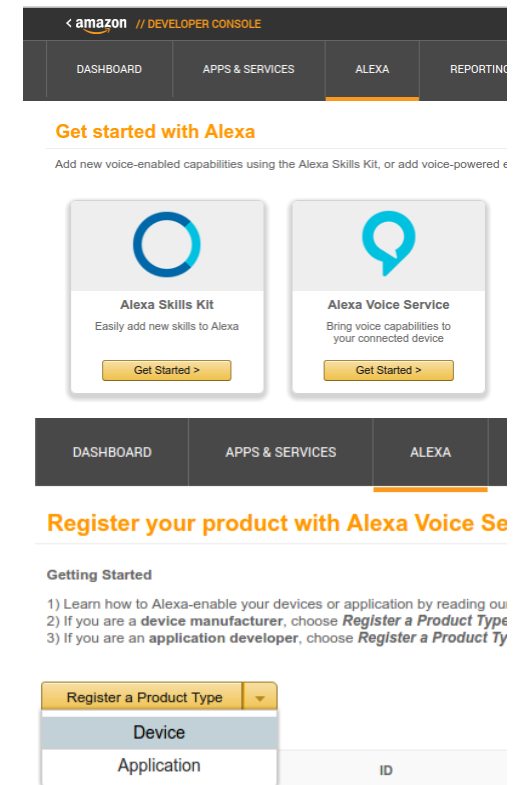
Source: pi-top.com

Clone the Alexa Sample Application

- ✱ The path of least resistance is to clone the Alexa sample application from Amazon's github repo
 - ▶ git clone <https://github.com/alexasampleapp/alexasampleapp>
- ✱ This will result in an alexa-avs-sample-app folder on your media

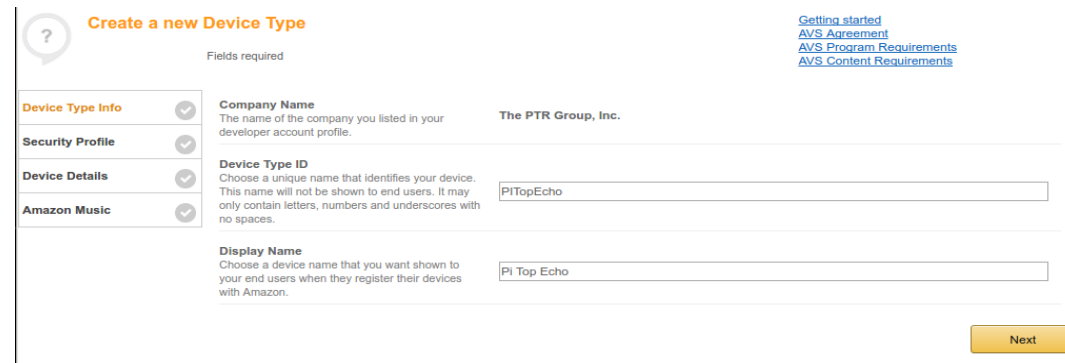
Register Your Device

- ✦ We next need to register our device with Amazon in order to get authentication codes to access AVS
- ✦ From the Developer Console, select “Alexa”
- ✦ Then, under Alexa Voice Service, select “Get Started”
- ✦ From this page, select “Register a Product Type” and choose “Device”



Register Your Device #2


- ✦ Enter a device ID and display name and select “Next”
 - ▶ I went with “PiTopEcho” and “Pi Top Echo”
- ✦ This will take you to the Security Profile screen
 - ▶ Use the pull down to select “Create a new profile”
- ✦ Fill out the “Security Profile Name” and “Security Profile Description” fields and select “Next”
 - ▶ This will display your security credentials for services like Amazon Music (copyright tracking) as (almost) seen on the next page



The screenshot shows a web form titled "Create a new Device Type" with a question mark icon. On the right, there are links: "Getting started", "AVS Agreement", "AVS Program Requirements", and "AVS Content Requirements". The form is divided into sections with checkboxes: "Device Type Info", "Security Profile", "Device Details", and "Amazon Music", all of which are checked. The "Company Name" field is filled with "The PTR Group, Inc.". The "Device Type ID" field is filled with "PiTopEcho". The "Display Name" field is filled with "Pi Top Echo". A "Next" button is located at the bottom right.


Fields required	Device Type Info	Security Profile	Device Details	Amazon Music
Company Name	The name of the company you listed in your developer account profile.	The PTR Group, Inc.		
Device Type ID	Choose a unique name that identifies your device. This name will not be shown to end users. It may only contain letters, numbers and underscores with no spaces.	PiTopEcho		
Display Name	Choose a device name that you want shown to your end users when they register their devices with Amazon.	Pi Top Echo		


Register Your Device #3


 **Create a new Device Type**


Fields required

[Getting started](#)
[AVS Agreement](#)
[AVS Program Requirements](#)
[AVS Content Requirements](#)


Device Type Info 

Security Profile 

Device Details 

Amazon Music 


You need a security profile to identify your device. Your security profile credentials - client ID and client secret - allow your device to securely identify itself to the Alexa Voice Service. If you are building a website, click here to [Learn More](#). If you are building an Android or iOS app, click here to [Learn More](#).


Security Profile  A security profile is how Amazon identifies your device. PiTopEcho Edit

General Web Settings Android/Kindle Settings iOS Settings

Security Profile Description
Choose a description for your security profile for Amazon services to use in communicating with you. Pi Top Echo


Security Profile ID
This ID will identify your security profile in Amazon services. amzn1.application.

Client ID  This is a value specific to you that is assigned to you when you register with Login with Amazon. amzn1.application-oa2-client.

Client Secret  This is a secret specific to you that is assigned to you when you register with Login with Amazon. Confidential.


Successfully created security profile.

Next

 Select "Next"

ELC-Portland-0217-18

Copyright 2017, The PTR Group, Inc.

 **PTR**

Register Your Device #4

[< Back to the list](#)




Create a new Device Type

Fields required

Device Type Info	✓
Security Profile	✓
Device Details	✓
Amazon Music	✓

Image
Upload an image sized 142(width)x130(height) pixels in either PNG or JPG format. This image is displayed on your customer's [Manage Your Content and Devices](#) page.



Category
Choose the category that best describes where and how your device is used.

Description
Please provide a brief description of your device and its functionality.

Do you have plans to make your product available to the general public?

☐ Yes ☒ No

Is your device a children's product or is it otherwise directed to children under the age of 13?

☐ Yes ☒ No

Not sure? [Learn more](#)

✖ Click “Next”

Register Your Device #5

✱ If you want your device to have access to Amazon Music, then you'll need to fill out another form and submit it for approval

✱ You can skip this step if you're not interested in Amazon Music services

Apply for access to Amazon Music? Yes No
Amazon Music is currently limited to Alexa-enabled products approved for commercial distribution and use. If you wish to apply, please answer the questions below and submit your request.

Note: Amazon Music isn't required to access other Alexa skills.

My device supports both HTTP Live Streaming (HLS) and progressive download as means to render music playback. ☒

My device will not allow offline playback of Amazon Music content. ☒

Please briefly describe your device's buffering/caching mechanisms.

My device will not incorporate Amazon Music into a game experience. ☒

My device will not enable synchronization of Amazon Music with video, photos or other visual media, e.g. a slideshow synchronized with Amazon Music. ☒

Please briefly describe your device's music playback capabilities, e.g., speaker quality, audio output, Bluetooth support, etc.

By submitting this form, you agree to [Amazon Voice Service Agreement](#).

Security Redirect URLs

✖ There's one more thing you need to do with the device security settings and that's to set up the security redirect URLs

✖ You'll need:

- ▶ Allowed Origins: `https://localhost:3000`
- ▶ Allowed Return URLs: <https://localhost:3000/authresponse>

✖ This will allow your local device to talk to the security manager at Amazon

Redirect URLs



Pi Top Echo

Fields required

[Getting started](#)
[AVS Agreement](#)
[AVS Program Requirements](#)
[AVS Content Requirements](#)

Device Type Info	✓
Security Profile	✓
Device Details	✓
Amazon Music	✓

You need a security profile to identify your device. Your security profile credentials - client ID and client secret - allow your device to securely identify itself to the Alexa Voice Service. If you are building a website, click here to [Learn More](#). If you are building an Android or iOS app, click here to [Learn More](#).

Security Profile ?

A security profile is how Amazon identifies your device.

PiTopEcho ▼

Edit

General

Web Settings

Android/Kindle Settings

iOS Settings

Allowed Origins ?

Your website origin, when using Login with Amazon.

https://localhost:3000

Allowed Return URLs ?

If you make HTTPs calls to Login with Amazon with redirect_uris, specify them here.

https://localhost:3000/authresponse

https://localhost:3000/authresponse

Save

Successfully updated security profile.

Next

Transfer Your Credentials

- ✱ Go back to the Security Profile tab and copy the Product ID, Client ID and Client Secret into the `automated_install.sh` in the Alexa sample app you downloaded earlier
 - ▶ There are links in the script file to help
- ✱ Fill out the rest of the script with your information as desired
- ✱ Next, run the `automated_install.sh` script
 - ▶ The installation process will take about 30 minutes on the Pi and you'll be asked several questions up front and then a lot of repository code will be pulled down
 - Fast Internet links are good!

Running the Service

- ✱ Once the software is configured and installed, we run the services needed to get to AVS
 - ▶ The web service, the sample app and the wake word engine
- ✱ For testing purposes, we can run these in separate windows for now
 - ▶ We'll run them in init scripts when we're done
- ✱ In the first window, go into your alexa-avs-sample-app/samples directory and cd into **companionService** and run **npm start**
 - ▶ This will start the web service

In the Second Window...

✱ Open a second window and navigate to:

- ▶ `<alexa-avs-sample-app>/samples/javaclient`

✱ Run

```
$ mvn exec:exec
```

- ▶ This is the app that communicates with AVS

✱ This will open a dialog box that reads something like:

- ▶ "Please register your device by visiting the following URL in a web browser and following the instructions: <https://localhost:3000/provision/d340f629bd685dee...>
Would you like to open the URL automatically in your default browser?"
 - Select "Yes" to open the URL

✱ When your browser opens, you may get a warning that the connection is not private

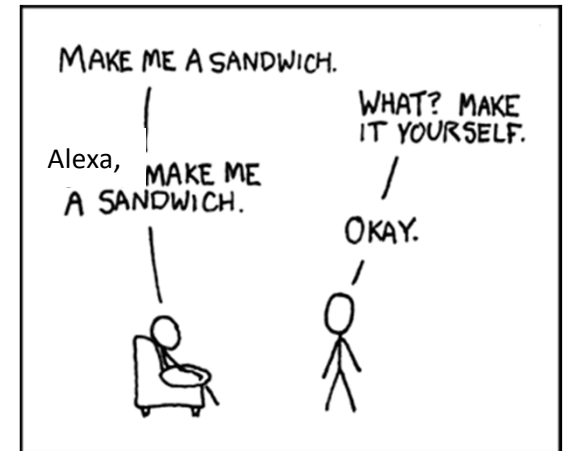
- ▶ Tell it to proceed anyway

In the Browser...

- ✱ You'll be directed to a page that asks you to log in to your Amazon account
- ✱ After logging in, you'll be directed to the "Developer Authorization" page confirming that you want your device to access the security profile you created earlier
 - ▶ Click "Okay"
- ✱ This should bring you to a page that says: "device tokens ready"
 - ▶ Minimize the browser and select OK on the dialog box

Starting the Wake Word Engine

- ✱ In the third window, we'll run the KITT.AI wake word engine
 - ▶ There are 2 that are supported, but KITT.AI seems to be the better of the two
- ✱ In the third window, navigate to `<alexa-avs-sample-app>/samples/wakeworkAgent/src`
- ✱ Start the wake word engine of choice:
`$./wakeworkAgent -e kitt_ai`
- ✱ With the wake word agent running, test out the system by saying “Alexa” and ask a test question
- ✱ If everything worked, you have Alexa working!
 - ▶ Now, it's time to button it up



Source: xkcd.com

Making it Standalone

- ✖ To simplify settings, install VNC

```
$ sudo apt-get install tightvncserver
```

- ▶ Then, run **tightvncserver**

- ✖ Set up VNC to launch every time we start the device

```
$ cd /home/pi
```

```
$ cd .config
```

```
$ mkdir autostart
```

```
$ cd autostart
```

```
$ vi tightvnc.desktop ; use your favorite editor here
```

- ✖ Add the following lines and save the file:

```
[Desktop Entry]
```

```
Type=Application
```

```
Name=TightVNC
```

```
Exec=vncserver :1
```

```
StartupNotify=false
```

Making it Standalone #2

- ✦ Depending on how you want to work, you can create a script that will open the three windows and start the service from VNC
- ✦ Or, you can create another autostart function like we did for tightvncserver

Adding Skills

- ✦ In order to enable skills, we'll need to install the Amazon Echo app on our smartphone
- ✦ Once you sign in on your amazon account, you should see the device you just set up
- ✦ Access the “Skills” section of the app to browse and enable skills for your device
- ✦ Sit back and order a pizza... 😊

Summary

- ✱ In this session, we had a whirlwind tour of adding voice services to an embedded Linux device
 - ▶ Works with both ARM and x86-based platforms
- ✱ This heavily leveraged the sample code and set up from Amazon
 - ▶ Once the code is running, feel free to register additional devices and start making new skills
- ✱ There are tutorials on Amazon's site for new skills
- ✱ The system is only as good as its audio processing
 - ▶ We need to extend the microphone with an audio beam former so we can have better echo cancellation and a steerable microphone array
 - Something for next ELC maybe 😊
- ✱ Good Luck!